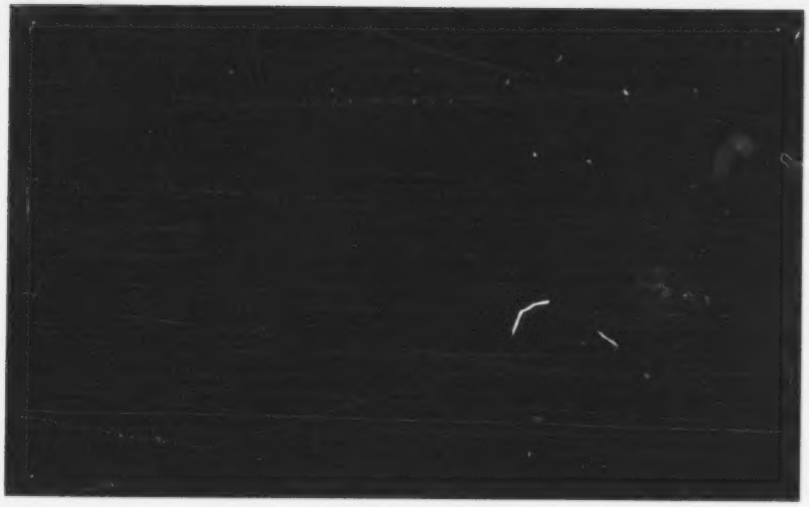
CENTRE DE RECHERCHE
SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH
ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE



CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A.
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: 05-24146 BIBPOLYTEC
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

CRT-729

A PARALLEL ROUTE BUILDING ALGORITHM FOR THE VEHICLE ROUTING AND SCHEDULING PROBLEM WITH TIME WINDOWS

by

Jean-Yves Potvin
Jean-Marc Rousseau

Centre de recherche sur les transports - Publication #729
October 1990

91-00966/

Dépôt légal - 4e trimestre 1990

Résumé. Dans cet article, nous décrivons un algorithme d'insertion pour la résolution du problème de tournées et horaires de véhicules avec fenêtres de temps. Cet algorithme construit les routes en parallèle et utilise une mesure de regret généralisé afin de sélectionner le prochain client à insérer. Des résultats numériques sont présentés sur l'ensemble des problèmes-tests de Solomon et des comparaisons sont faites avec son algorithme séquentiel.

Mots Clés. Transport, Tournées et Horaires, Fenêtres de temps, Heuristiques

Abstract. This paper describes an insertion algorithm for the Vehicle Routing and Scheduling Problem with Time Windows (VRSPTW). This algorithm builds routes in parallel and uses a generalized regret measure over all unrouted customers in order to select the next candidate for insertion. Numerical results on the standard set of problems of Solomon are reported as well as comparisons with his sequential algorithm.

KeyWords. Transportation, Vehicle Routing and Scheduling, Time Windows, Heuristics.

Section 1. Introduction

A very important aspect of any distribution system is the routing and scheduling of vehicles through a set of customers that require some kind of services. In fact, effective and efficient distribution management has been shown to be one of the keys for reducing costs and improving profits.

The Vehicle Routing Problem (VRP) is concerned with the design of minimum cost routes, starting and ending at a central depot, for a fleet of vehicles that services a set of customers with known demands. Each customer must be serviced exactly once, and the customers are assigned to vehicles so that the total demand on a route does not exceed vehicle capacity. This problem has been extensively studied in the literature and we refer to Bodin et al. (1983) for a comprehensive survey.

In the vehicle routing and scheduling problem with time windows (VRSPTW), these issues must be addressed with the additional complexity of allowable delivery times or time windows at each

customer. Moreover, a time window is introduced at the depot location so as to define a "scheduling horizon". Each vehicle's tour must start and end within the bounds of that horizon. It is thus acting a little bit like a capacity constraint (e.g. by reducing the scheduling horizon, only a few customers can be serviced by the same vehicle).

In the presence of time constraints, the total routing and scheduling costs include not only the total travel distance (or total travel time), but also the waiting time incurred when a vehicle arrives too early at a customer location. Hence, spatial and temporal aspects of the problem must be considered in order to identify good solutions.

Recently, the VRSPTW has emerged as an important area for progress and has been used to model many realistic applications. An interesting survey may be found in Solomon and Desrosiers (1988). Of particular interest is a recent paper by Solomon describing a variety of route construction heuristics for the VRPSTW (Solomon 1987). The results of his extensive computational study shows that a sequential time-space based insertion algorithm, which is a generalization of the Mole and Jameson's approach to the VRP (1976), has outperformed all other approaches on a set of routing test problems initially proposed by Christofides et al. (1979). Several authors, like Sorensen (1986), have confirmed the success of this approach. The aim of this paper is thus to show how the sequential insertion algorithm of Solomon can be substantially improved, for some problem instances, by using a parallel route building philosophy.

In the following, section 2 first introduces Solomon's approach to the VRSPTW problem. Section 3 then describes our parallel approach to the problem and introduces a new metric for insertion, which is based on a generalized regret measure over all unrouted customers. Finally, computational results are presented in section 4 and concluding remarks are made.

Section 2. A Sequential Approach to the VRSPTW

In this section, we introduce Solomon's sequential approach to the VRSPTW (Solomon 1987). Here, routes are created one at a time. A route is first initialized with a "seed" customer and the remaining unrouted customers are added to this route until it is full with respect to the scheduling horizon and/or capacity constraint. If unrouted customers remain, the initialization and insertion procedures are then repeated until all customers are serviced.

After initializing the current route with a seed customer, the method uses two criteria $c_2(i,u,j)$ and $c_1(i,u,j)$ to respectively select and

insert customer u between adjacent customers i and j in the current partial route. Let $(i_0, i_1, i_2,..., i_m)$ be the current route with i_0 and i_m representing the depot. For each unrouted customer, we first compute its best feasible insertion place in the emerging route as:

$$c_1^*(i(u),u,j(u)) = \min_{p=1,...,m} [c_1(i_{p-1},u,i_p)].$$

Of course, inserting u between i_{p-1} and i_p in the current route can potentially introduce a delay for service at all following customers $(i_p,...,i_m)$. Hence, the service time at any of these customers can now exceed their time window's upper bound or, alternatively, the route can now extend beyond the scheduling horizon. A feasibility check is thus made each time an insertion is considered and the cost $c_1(i_{p-1},u,i_p)$ is set to an arbitrary large value if this check fails. The reader is referred to Solomon's paper for an efficient way of doing such a feasibility check (Solomon 87).

Next, the best customer u to be inserted in the route is selected as the one for which:

$$c_2^*(i(u^*),u^*,j(u^*)) = \text{optimum}_u[c_2(i(u),u, j(u))], \quad u \text{ unrouted and feasible.}$$

Client u^* is then inserted in the route between $i(u^*)$ and $j(u^*)$. When no more customer with feasible insertions can be found, the method starts a new route, unless it has already routed all customers.

In his paper, Solomon describes three different heuristics based on this generic approach and the one that outperformed all others has the following characteristics:

1. Initialization is done with the farthest unrouted customer from the depot or the customer with the earliest time window's upper bound (earliest deadline).

2. Insertion and selection costs are the followings.

   We first consider the minimum insertion costs for each unrouted customer $u \in S_u$:

   $$c_1^*(i(u),u,j(u)) = \min_{p=1,...,m} [c_1(i_{p-1},u,i_p)],$$

   $$c_1(i,u,j) = \alpha_1 \cdot c_{11}(i,u,j) + \alpha_2 \cdot c_{12}(i,u,j), \quad \alpha_1+\alpha_2=1, \ \alpha_1 \geq 0, \ \alpha_2 \geq 0.$$

   where

   $$c_{11}(i,u,j) = d_{i,u} + d_{u,j} - d_{i,j}, \quad d_{k,l} = \text{distance (in time units) between k and l.}$$

$c_{12}(i,u,j) = b_{u,j} - b_j$, b_k = current service time at k.
$b_{k,l}$ = new service time at l, given that k is now on the route.

Then, select u* such that:

$$c_2(i(u^*),u^*,j(u^*)) = \max_{u \in S_u} [c_2(i(u),u,j(u))]$$
$$c_2(i(u),u,j(u)) = \beta \cdot d_{depot,u} - c_1^*(i(u),u,j(u)), \quad \beta \geq 0$$

and insert u* between i(u*) and j(u*).

As we can see, the insertion cost $c_1(i,u,j)$ for customer u is a combination of extra-mileage (expressed in time units) and time delay. Hence, insertion of customer u takes into account spatial and temporal aspects of the problem. Solomon's heuristic selects the customer which maximizes criterion $c_2(i,u,j)$. With $(\beta,\alpha_1,\alpha_2,) = (2,1,0)$, this is the maximum benefit derived from servicing a customer on the partial route being constructed rather than on a direct route. This measure is thus derived from the saving metric of Clarke and Wright (1964).

Solomon applies this algorithm with the two initialization criteria mentioned above and with four distinct parameter settings: $(\beta, \alpha_1, \alpha_2,) = (1,1,0)$, $(2,1,0)$, $(1,0,1)$ and $(2,0,1)$. The algorithm is thus applied eight times to each problem and the best overall solution is selected. A complete description and a flowchart for this algorithm may be found in Appendix 1, at the end of the paper.

Experiments that we ran with Solomon's sequential algorithm demonstrate that, in some cases, the last unrouted customers tend to be widely disseminated over the geographic area. When this happens, the last routes are usually of very poor quality, and it can even take too many vehicles to service those last remaining customers. This is a well known intrinsic weakness of the myopic sequential approaches. In order to alleviate this problem, we propose instead a parallel approach for initializing and building the set of routes. This new algorithm is described in the next section.

Section 3. A Parallel Approach to the VRSPTW

The new parallel algorithm makes use of the generic framework of Solomon, but introduces new insertion and selection costs as well as a parallel route building philosophy. Here a set of n_r routes is initialized at once, and the remaining customers are then inserted into any one of those routes.

One difficulty with the parallel approach, however, is to determine what the initial number of routes n_r should be. In order to do that, we estimate n_r by applying Solomon's sequential algorithm only once, using the farthest customer from the depot for initialization and the following single parameter setting: $(\beta,\alpha_1,\alpha_2) = (1,1,0)$. In this way, we obtain an initial estimate for n_r and we can build n_r routes in parallel.

The generic insertion framework of Solomon is then used with the following specific characteristics:

1. Initialization is done by selecting n_r seeds among all customers. This selection process is done sequentially, and with respect to the previously selected seeds. The rule for selecting the next seed is the following: select the customer which is the farthest from the depot and from all those customers that have already been selected. Assuming that S_{seed} is the set of currently selected seeds (including the depot) and S_u is the set of remaining customers, the rule for selecting the next seed u^* is:

   select u^* such that $\min_{s\in S_{seed}} d_{s,u^*} = \max_{u\in S_u} \{\min_{s\in S_{seed}} d_{s,u}\}$.

   In this way, we get a good initial distribution of all seeds over the geographic area and the insertion of the remaining customers is less likely to leave the last unrouted customers widely apart.

2. The insertion and selection costs are the followings:

   We first consider the minimum insertion costs for each unrouted customer $u\in S_u$ in each one of the n_r initial routes:

   $$c_{1r}^*(i_r(u),u,j_r(u)) = \min_{p=1,\ldots,m} [c_{1r}(i_{r_{p-1}},u,i_{r_p})], \quad r=1,\ldots,n_r.$$

   $$c_{1r}(i_r,u,j_r) = \alpha_1 \cdot c_{11r}(i_r,u,j_r) + \alpha_2 \cdot c_{12r}(i_r,u,j_r), \quad \alpha_1+\alpha_2=1, \ \alpha_1\geq 0, \ \alpha_2\geq 0.$$

   where

   $$c_{11r}(i_r,u,j_r) = d_{i_r,u} + d_{u,j_r} - d_{i_r,j_r},$$

   $$c_{12r}(i_r,u,j_r) = b_{u,j_r} - b_{j_r},$$

   $d_{k,l}$ = distance (in time units) between k and l.

   b_k = current service time at k.

   $b_{k,l}$ = new service time at l, given that k is now on the route.

Then, select u* such that:

$$c_2(u^*) = \max_{u \in S_u} [c_2(u)]$$

$$c_2(u) = \sum_{r \neq r'} [c_{1r}^*(i_r(u),u,j_r(u)) - c_{1r'}^*(i_{r'}(u),u,j_{r'}(u))],$$

where

$$c_{1r'}^*(i_{r'}(u),u,j_{r'}(u)) = \min_{r=1,\ldots,n_r} [c_{1r}^*(i_r(u),u,j_r(u))]$$

and insert u* between $i_{r'}(u^*)$ and $j_{r'}(u^*)$.

As we can see from the definition of $c_2(u)$, a generalized regret measure over all routes is used for selecting the next customer to be inserted. Since $c_{1r}^*(i_r(u),u,j_r(u))$ is set to an arbitrary large value when there is no feasible insertion point within route r for customer u, the generalized regret measure can foresee difficulties to come. In particular, a customer's measure increases by some arbitrary large value whenever a route becomes non feasible for that customer because of the insertion of another customer (unless all routes become non feasible, in which case the metric equals 0). Therefore, the generalized regret measure partitions the set of unrouted customers based on the number of routes that remain feasible for each one of them. For example, if we assume that the arbitrary large value is L, the regret measure of all unrouted customers with n alternative feasible routes $1 <= n <= n_r$ will be in the neighborhood of $L(n_r - n)$. Of course, it is important to take care first of customers with few alternative feasible routes, that is, customers with a high regret measure. When the number of feasible routes is the same for two customers, the regret measure then discriminates them by looking at the best insertion points within their feasible routes. This approach is, in fact, a heuristic way of solving a generalized assignment problem with time constraints.

This parallel algorithm is applied to each problem with the parameter settings $(\alpha_1, \alpha_2) = (0.5, 0.5), (0.75, 0.25), (1.0, 0.0)$, and the best overall solution is chosen. It must however be noticed that the algorithm is applied more than three times to each problem because we also look for potential feasible solutions with a smaller number of routes than the one given by Solomon's initial estimate (that is, this estimate is used only as a starting point).

This is done by stopping the procedure as soon as a feasible solution is found for a given number of routes (initially, Solomon's estimate) and a given parameter setting. We then remember the parameter settings that have not yet been tried for the current number of routes (if any), we decrease the number of routes by one

and we repeat the whole procedure with $n_r = n_r - 1$ routes. The number of routes is reduced in this way until we get some value n_{rinf} for which no feasible solution is found. At that point, we simply backtrack to $n_r = n_{rinf} + 1$ routes and go on with the remaining parameter settings (if any) for that number of routes. The best solution over the three parameter settings for $n_r = n_{rinf} + 1$ is then selected. A more complete description and a flowchart for this algorithm may be found in Appendix 2 at the end of the paper.

In order to evaluate the potential of the parallel approach, as compared to the sequential approach, we performed a number of computational tests based on Solomon's standard set of problems. Those results are presented in the next section.

Section 4. Computational Results

4.1 The set of problems

For the computational tests, we used the standard set of problems of Solomon. The design of those test problems highlights factors that can affect the behavior of routing and scheduling heuristics, like geographical data, number of customers serviced by a vehicle and time window characteristics (e.g. percentage of time-constrained customers, tightness and positioning of the time windows, width of the time horizon). The data used for customer coordinates and demands are based on the data from the standard set of routing problems of Christofides et al. (1979). The geographical data are either randomly generated with a uniform distribution (problem sets R1 and R2), clustered (problem sets C1 and C2) or semi-clustered with a mix of randomly distributed and clustered customers (problem sets RC1 and RC2). Problem sets R1, C1 and RC1 have a short scheduling horizon, and allow only a few customers per route. Conversely, problem sets R2, C2 and RC2 have a long scheduling horizon and allow a larger number of customers per route. More details about these problems, in particular details relating to the generation of the time windows, may be found in Solomon's paper (1987).

All the test problems are 100-customer euclidean problems. Service times are 10 units per customer for problem sets R1, R2, RC1 and RC2, and 90 units per customer for problem sets C1 and C2. Finally, a homogeneous fleet is assumed.

In section 4.2, we compare results that were obtained with the parallel algorithm and Solomon's sequential algorithm, without any post-optimization procedures (as in Solomon's original paper). Section

4.3 then gives some insight about solution quality with respect to optimum solutions, with and without post-optimization.

4.2 Sequential vs Parallel approach

For these experiments, we ran Solomon's original code as well as our Fortran implementation of the parallel algorithm on an IBM-compatible microcomputer. Table 1 compares Solomon's sequential approach (SEQUENTIAL) with the parallel approach (PARALLEL). For each type of problems, we show the average number of routes, total route time, and computation time (in seconds). The total route time is the sum of total travel time, waiting time and service time.

For Solomon's algorithm, we also show within parentheses the results as reported in his original paper. The slight differences between his results and ours stem from the distinct hardware implementations (in particular, our microcomputer's floating point precision is not the same as the DEC-10 used by Solomon, and we observed on a small number of problems that this fact was responsible for slightly different solutions).

For comparison purposes, solution quality is based first on the number of routes, then on total route time.

R1 12 problems	Number of routes	Route time	Computation time (sec.)
SEQUENTIAL	13.5 (13.6)	2688.5 (2695.5)	543.5
PARALLEL	13.8	2736.1	911.1

R2 11 problems	Number of routes	Route time	Computation time (sec.)
SEQUENTIAL	3.3 (3.3)	2555.0 (2578.1)	3332.6
PARALLEL	3.3	2526.3	1927.6

Table Ia Random problems

C1 9 problems	Number of routes	Route time	Computation time (sec.)
SEQUENTIAL	10.0 (10.0)	10094.0 (10104.2)	577.7
PARALLEL	10.9	10991.6	901.3

C2 8 problems	Number of routes	Route time	Computation time (sec.)
SEQUENTIAL	3.1 (3.1)	9918.3 (9921.4)	1858.4
PARALLEL	3.75	10426.1	1253.8

Table Ib Clustered Problems

RC1 8 problems	Number of routes	Route time	Computation time (sec.)
SEQUENTIAL	13.5 (13.5)	2770.4 (2775.0)	483.4
PARALLEL	13.35	2879.4	862.7

RC2 8 problems	Number of routes	Route time	Computation time (sec.)
SEQUENTIAL	4.0 (3.9)	2982.2 (2955.4)	2574.5
PARALLEL	3.5	2806.6	1438.9

Table Ic Mixed Problems

From the results shown in Table I, we can draw the following conclusions:

a) For randomly generated problems, the two approaches performed equally well. On problem set R1, the sequential approach slightly outperformed the parallel approach with respect to both the number of routes and the total route time. However, on problem set R2, the number of routes is the same for both approaches, and the parallel approach is slightly better with respect to total route time.

b) For clustered problems C1 and C2, the sequential approach outperformed the parallel approach. This is not too surprising, since a sequential approach is well adapted to such purely clustered problems. We can describe its general behavior in the following way: first, it selects a seed customer in one cluster and "empties" it during the insertion phase. Then, it initializes a second route by selecting the seed customer in another cluster and empties it again. This process is repeated until all clusters are done.

   The parallel approach, on the other hand, is more prone to select initial seed customers in the same cluster, since they are all selected at once at the beginning of the procedure. This phenomenon underlines the paramount importance of a clever initialization procedure.

c) For mixed problems RC1 and RC2, the parallel approach outperformed the sequential approach. On problem set RC2, in particular, we were able to reduce the total number of routes by 12% and the total route time by 6%. If we consider that a fixed number of 10 time units per customer is included for service time (for a total of 1000 time units), the improvement with respect to travel time and waiting time is really in the order of 9%, which is quite substantial. Those results are the best reported so far on that set of problems. For problem set RC1, the parallel approach still outperformed the sequential approach, and allowed a slight reduction in the total number of routes.

The parallel approach thus looks very interesting for mixed random-clustered problems. Since real-life problems seldom are purely random or clustered, and usually show more or less well-defined clustered patterns, we feel that our algorithm is an interesting alternative to the classical algorithm of Solomon for such VRSPTW problems.

4.3 Comparisons with optimum solutions

The second set of experiments is aimed at evaluating the accuracy of the above solutions with respect to optimality. To this end, we compared the parallel and sequential algorithms with solutions generated by the column generation technique of Desrosiers et al. (1984). Since the complexity of this algorithm grows exponentially with the size of the problem, only a few problems from Solomon's set have been solved optimally. Unfortunately, no optimum solutions are known for the sets of problems RC1 and RC2, for which the parallel approach performed particularly well. The only known optimum solutions are for problems R101 and R102 in set R1, and C101, C102, C106, C107, C108 in problem set C1. Time window's characteristics of those problems are shown in Table II.

Problem	% of customers with time windows	Average width of time windows
R101	100	10.0
R102	75	10.0
C101	100	60.76
C102	75	61.27
C106	100	156.15
C107	100	180.00
C108	100	243.28

Table II Characteristics of Problems

The computational results are shown in Table IIIa and IIIb. The first two colums of Table IIIa and IIIb are the results respectively obtained with the parallel and sequential algorithms, before and after an Or-Opt post-optimization procedure (Or 1976). Optimum solutions are shown in the last column of both tables. The first number in each cell is the total number of routes and the second number is the total travel time.

Problem	PARALLEL	PARALLEL & OR-OPT	OPTIMUM
R101	21 1915.6	21 1769.0	18 1607.0
R102	18 1706.5	18 1550.4	17 1434.0
C101	12 1876.9	11 1239.7	10 827.3
C102	11 1318.8	11 1242.2	10 827.3
C106	11 1267.3	11 1015.2	10 827.3
C107	11 1784.0	11 1158.0	10 827.3
C108	11 1303.3	11 1028.6	10 827.3

Table IIIa Parallel solutions vs Optimum solutions

Problem	SEQUENTIAL	SEQUENTIAL & OR-OPT	OPTIMUM
R101	21 1826.0	21 1693.3	18 1607.0
R102	19 1711.0	19 1520.2	17 1434.0
C101	10 851.4	10 827.3	10 827.3
C102	10 964.6	10 833.6	10 827.3
C106	10 881.0	10 827.3	10 827.3
C107	10 902.4	10 827.3	10 827.3
C108	10 853.1	10 827.3	10 827.3

Table IIIb Sequential solutions vs Optimum solutions

Although such results are not conclusive, we can see that Solomon's algorithm performed very well on clustered problems. It found the minimum number of routes in each case and reached

optimum (after post-optimization) for problems C101, C106, C107 and C108. This supports the previous claim about the power of Solomon's sequential approach for purely clustered problems.

The results are not so good, however, for purely random problems. For both problems R101 and R102, the parallel and sequential algorithms did not find the minimum number of routes. After post-optimization, however, we can see that total travel time is not far from optimum. Those results indicate that when some randomness is introduced into the distribution of the customers, the parallel and sequential algorithms do not perform so well with respect to optimality. Hence, there is still room for improvement for such problems and this improvement could perhaps be achieved through more clever initialization procedures for both the sequential and parallel approaches.

Section 5. Concluding Remarks

This paper has described a parallel route building procedure for vehicle routing problems with time windows. Computational experiments have shown that the parallel approach is better than the sequential approach for problems with a mix of randomly distributed and clustered customers. The parallel approach compares with Solomon's algorithm on pure random problems and is outperformed on pure clustered problems. The parallel approach thus looks interesting, in particular for real-life problems which seldom are purely random or clustered problems. Comparisons with optimum solutions have also shown that Solomon's algorithm is quite powerful for purely clustered problems. However, this is not true anymore when some randomness is introduced into the distribution of the customers, and interesting research avenues are opened for improving upon the parallel or sequential algorithms in such contexts.

Acknowledgements. We would like to thank Lucie Bibeau and Patrice Bergeron for the hours spent at running and analyzing the computational experiments. Thanks also to Jacques Desrosiers and Martin Desrochers who gave us the code and the set of problems of Marius M. Solomon, as well as known optimum solutions. Finally, this research would not have been possible without the financial support of the Natural Sciences and Engineering Council of Canada (NSERC).

References

Bodin L., Golden B.L., Assad A. and Ball M. (1983), "Routing and Scheduling of Vehicles and Crews: The State of the Art", *Computers and Operations Research 10*, 62-212.

Christofides N., Mingozzi R. and Toth P. (1979), "The Vehicle Routing Problem", in *Combinatorial Optimization*, N. Christofides, R. Mingozzi, P. Toth and C. Sandi (eds), John Wiley and Sons, New-York.

Clarke G. and W. Wright (1964), "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", *Operations Research 12*, 568-581.

Desrosiers J., Soumis F. and Desrochers M. (1984), "Routing with Time Windows by Column Generation", Networks 14, 545-565.

Mole R., Jameson S. (1976), "A Sequential Route-Building Algorithm Employing a Generalized Savings Criterion", *Operational Research Quarterly 27*, 503-511.

Or I. (1976), "Traveling Salesman-type Combinatorial Problems and their relation to the Logistics of Blood Banking", Ph.D. Thesis, Dept. of Industrial Engineering and Management Sciences, Northwestern University.

Solomon M.M. (1987), "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints", *Operations Research 15 (2)*, 254-265.

Solomon M.M. and Desrosiers J. (1988), "Time Window Constrained Routing and Scheduling Problems", *Transportation Science 22 (1)*, 1-13.

Sorensen B. (1986), "Interactive Distribution Planning", Ph.D. Dissertation, IMSOR, Lyngby, Denmark.

Appendix 1. The sequential tour building procedure

Variables and constants:

n_r = number of routes
t_{ij} = travel time between customers i and j.
b_j = service time at customer j.
$b_{u,j}$ = service time at customer j, given that customer u is inserted just before j

detour(i,u,j) = $t_{iu} + t_{uj} - t_{ij}$ (i.e. detour in time units)
delay(i,u,j) = $b_{u,j} - b_j$

$c_1(i,u,j) = \alpha_1 \cdot \text{detour}(i,u,j) + \alpha_2 \cdot \text{delay}(i,u,j)$
$c_1^*(i(u),u,j(u)) = \min_{i,j} c_1(i,u,j)$, where i and j are adjacent customers in the current route.
$c_2(i(u),u,j(u)) = \beta \cdot t_{depot,u} - c_1^*(i(u),u,j(u))$

Algorithm:

Step 0. Initialize the set S of (initialization criterion, parameter setting) pairs with $S_{init} \times S_{(\beta,\alpha1,\alpha2)}$, where S_{init} = {farthest unrouted customer, unrouted customer with earliest deadline} and $S_{(\beta,\alpha1,\alpha2)}$ = {(1,1,0),(2,1,0),(1,0,1),(2,0,1)}.

Step 1. Select a remaining (initialization criterion, parameter setting) pair and remove it from S. The initialization criterion is used at Step 2 for initializing a new route with a seed customer and the parameter setting is used at Step 3 and Step 4 for the selection and insertion of the remaining unrouted customers. Set n_r to 1.

Step 2. Initialize route n_r by selecting a seed customer according to the initialization criterion.

Step 3. Evaluate insertion costs. For all unrouted customer u, evaluate insertion costs $c_1(i,u,j)$ for all adjacent customers i and j in route n_r and find $c_1^*(i(u),u,j(u))$. Notice that $c_1(i,u,j)$ is set to an arbitrary large value if the insertion is non feasible.

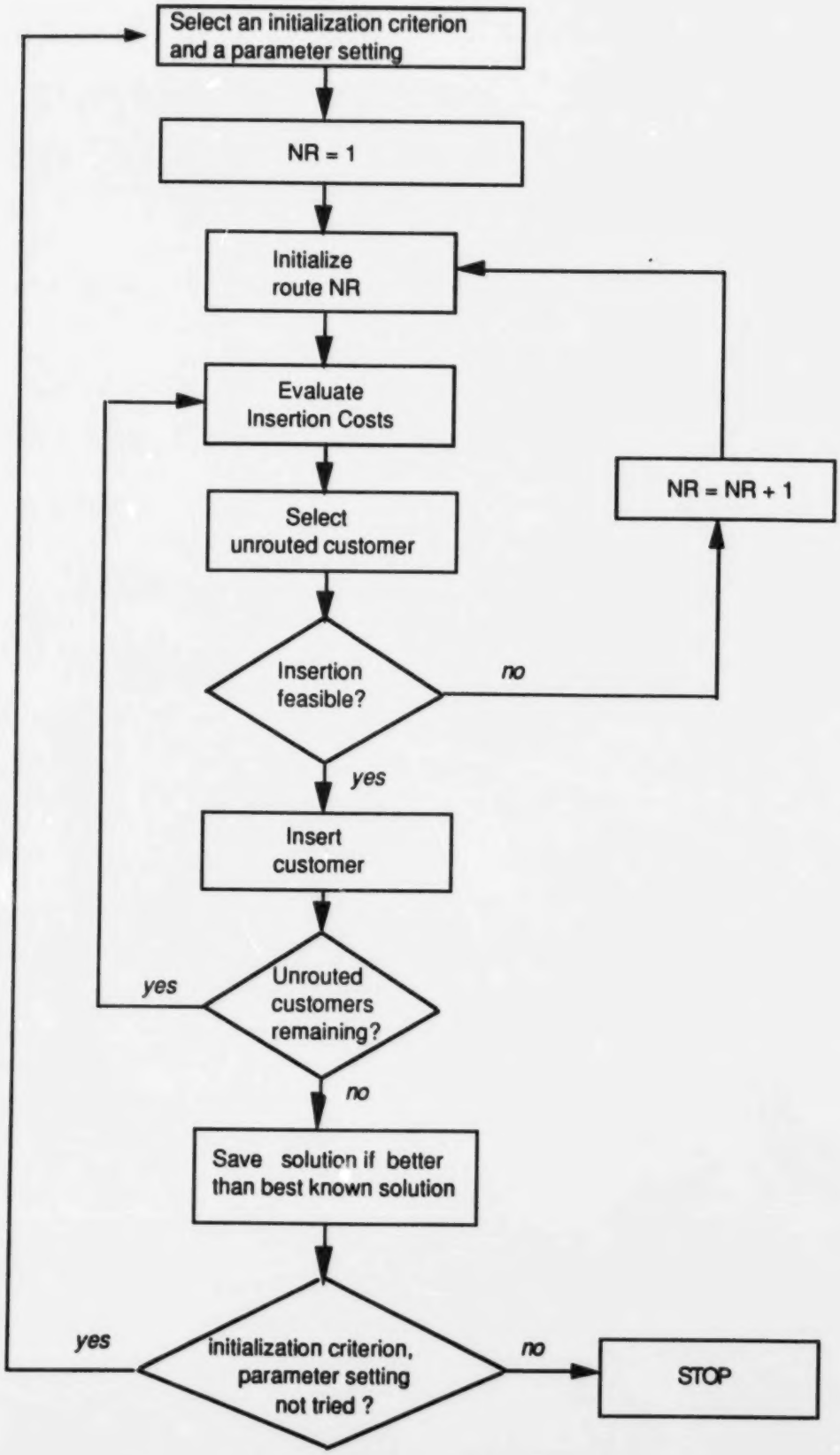
Step 4. Select the next customer u* to be inserted as the one for which $c_2(i(u),u,j(u))$ is maximum.

Step 5. Consider the insertion of customer u* in route n_r between i(u*) and j(u*).

If this insertion is feasible:
insert u*
if unrouted customers remain:
go to step 3,
otherwise:
go to step 6,
otherwise:
go to step 2 with $n_r = n_r+1$.

Step 6. Compare the current solution with the best known solution. The following criteria are considered in lexicographic order: total number of routes, total route time. If the current solution is better, remember it as the best known solution.

Step 7. If there are still (initialization criterion, parameter setting) pairs in S go to step 1, otherwise terminate with the best known solution.



Flowchart for the sequential algorithm

Appendix 2. The parallel tour building procedure

Variables and constants:

n_r = number of routes.
n_r^* = smallest number of routes.
t_{i_r,j_r} = travel time between customers i_r and j_r in route r.
b_j = service time at customer j.
$b_{u,j}$ = service time at customer j, given that customer u is inserted just before j

$detour(i_r,u,j_r) = t_{i_r,u} + t_{u,j_r} - t_{i_r,j_r}$ (i.e. detour in time units)
$delay(i_r,u,j_r) = b_{u,j_r} - b_j$

$c_{1r}(i_r,u,j_r) = \alpha_1 \cdot detour(i_r,u,j_r) + \alpha_2 \cdot delay(i_r,u,j_r)$, $r=1,...,n_r$
$c_{1r}^*(i_r(u),u,j_r(u)) = min_{i_r,j_r} c_{1r}(i_r,u,j_r)$, where i_r and j_r are adjacent customers on route r, r=1,...,nr.

$c_2(u) = \Sigma_{r\neq r'}[c_{1r}^*(i_r(u),u,j_r(u)) - c_{1r'}^*(i_{r'}(u),u,j_{r'}(u))]$
where
$c_{1r'}^*(i_{r'}(u),u,j_{r'}(u)) = min_r\, c_{1r}^*(i_r(u),u,j_r(u))$.

Algorithm:

Step 0. Estimate the number of routes n_r by applying Solomon's sequential algorithm with $(\beta,\alpha_1,\alpha_2) = (1,1,0)$. Set n_r^* to 0.

Step 1. Initialize the set of parameters $S_{(\alpha 1,\alpha 2),n_r}$ with {(0.5, 0.5), (0.75, 0.25), (1.0, 0.0)}.

Step 2. Select a parameter setting (α_1, α_2) and remove it from $S_{(\alpha 1,\alpha 2),n_r}$.

Step 3. Route initialization: for initializing the first route, select the farthest customer from the depot and call it u_1. Set S_{seed} to {depot, u_1}, remove u_1 from the set of unrouted customers S_u and for r=2,..,n_r do:

select u_r such that $min_{s\in S_{seed}} t_{s,u_r} = max_{u\in S_u} \{min_{s\in S_{seed}} t_{s,u}\}$
add u_r to S_{seed} and remove u_r from S_u.

Step 4. Evaluate insertion costs. For all unrouted customer u, evaluate insertion costs $c_{1r}(i_r,u,j_r)$ for all adjacent customers i_r and j_r in route r and find $c_{1r}*(i(u),u,j(u))$ for r = 1,...,nr. Notice that $c_{1r}(i_r,u,j_r)$ is set to an arbitrary large value if there is no feasible insertion points.

Step 5. Select the next customer u* to be inserted as the one for which $c_2(u)$ is maximum.

Step 6. Consider the insertion of customer u* between $i_{r'}(u^*)$ and $j_{r'}(u^*)$ in route r', where

$$c_{1r'}{}^*(i_{r'}(u^*),u^*,j_{r'}(u^*)) = \min_{r=1,\ldots,n_r} c_{1r}{}^*(i_r(u^*),u^*,j_r(u^*))$$

If this insertion is feasible:

insert u*;

if unrouted customers remain:

go to step 4,

otherwise

if $n_r = n_r{}^*$:

go to step 7,

otherwise:

remember the current solution as the best known solution and go to step 1 with $n_r = n_r{}^* - 1$.

otherwise:

if there are still parameter settings in $S_{(\alpha 1,\alpha 2),n_r{}^*}$:

go to step 2,

otherwise:

if $n_r = n_r{}^*$:

terminate with the best known solution,

otherwise:

set $n_r{}^*$ and n_r to $n_r + 1$;

if there are still parameter settings in $S_{(\alpha 1,\alpha 2),n_r{}^*}$:

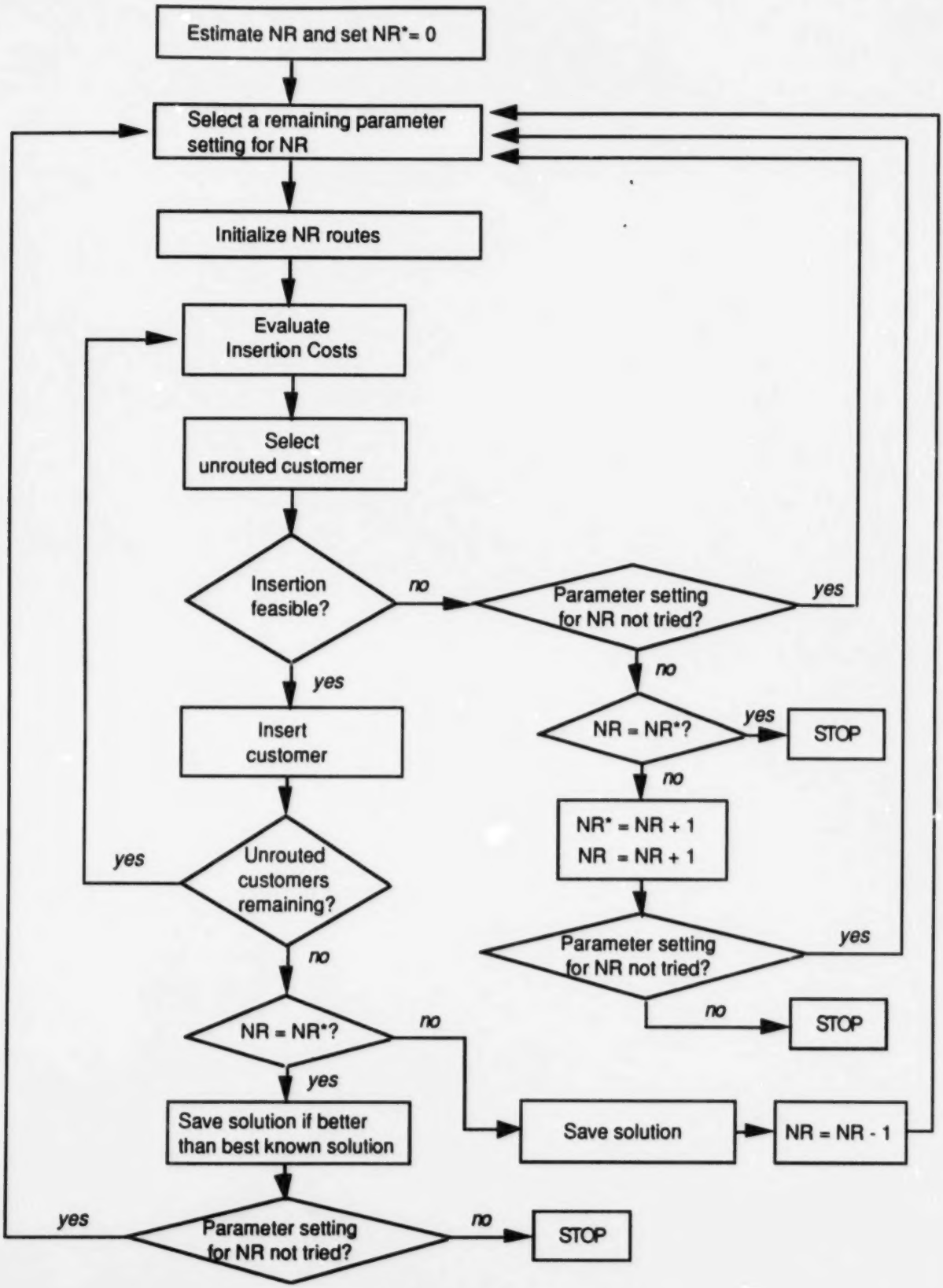
go to step 2,

otherwise:

terminate with the best known solution.

Step 7. Compare the current solution with the best known solution. The following criteria are considered in lexicographic order: total number of routes, total route time. If the current solution is better, remember it as the best known solution.

Step 8. If there are still parameter settings in $S_{(\alpha 1,\alpha 2),n_r*}$ go to step 2, otherwise terminate with the best known solution.



Flowchart for the parallel algorithm